L’entraînement de Monseigneur Node – Partie 3

version 5.0





1. Introduction

Monseigneur Node est très fier de son disciple, car même si vous avez eu des difficultés, vous les avez surmontées et vous n’avez pas abandonné. Vous voici arrivé dans la dernière partie dans laquelle vous allez pouvoir consolider vos acquis en créant un jeu des allumettes avec une intelligence artificielle.

A la fin de cette partie, vous serez fin prêt à conquérir le monde !

1. Consignes

* Pour ce projet-là, il vous sera demandé de créer un repository avec le nom : cc\_entrainement\_de\_monseigneur\_node.
* N’oubliez pas de push régulièrement !
* En cas de question, pensez à demander de l’aide à votre voisin de droite. Puis de gauche. Ou inversement. Puis demandez enfin à un Cobra si vous êtes toujours bloqué(e).
* Pensez à faire valider chaque partie que vous réaliserez à un Cobra lorsque vous l’aurez terminée.
* N’hésitez pas à faire des bonus et à ajouter des fonctionnalités lorsque votre projet sera terminé et validé.
* Vous avez tout à fait le droit d’utiliser internet pour trouver des réponses ou pour vous renseigner.

1. Récupération de ce que l’on écrit

Monseigneur Node aimerait vous instruire sur comment récupérer ce qu’il y a dans le terminal pendant l'exécution. Pour cela il faut utiliser le processus interne "process.stdin". C’est ce qui fera en sorte que l’on ait accès à ce qui est écrit dans le terminal lors de l’exécution d’un programme. Mais comment l’utiliser ?

Il faut attribuer ce processus à une variable, puis définir l’encodage des caractères, Monseigneur Node veut de [UTF-8](https://fr.wikipedia.org/wiki/UTF-8).

let entreeStandard = process.stdin;

entreeStandard.setEncoding("utf-8");

Si vous le souhaitez, vous pouvez ajouter une consigne pour savoir comment quitter l’exécution :

console.log("Dites quelque chose.\nDites \'exit\' pour quitter.");

Et maintenant mettez en route votre "entreeStandard" avec la méthode "on()". Celle-ci prend en paramètre un mode et une fonction anonyme. Pour que vous puissiez tester et comprendre un peu, voici un exemple :

entreeStandard.on("data", (data) => {  
 if (data === "exit\n" || data === "exit\r\n") {

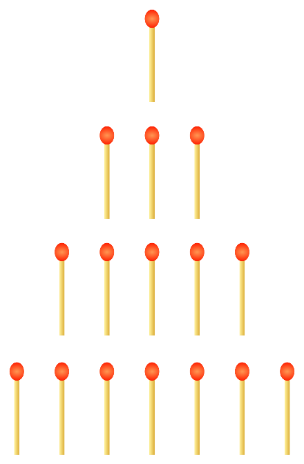
//on vérifie si l'utilisateur a entré le mot exit (+ '\n' car quand on appuie sur entrer ça l'ajoute automatiquement ou '\r\n' pour Windows)  
 console.log("Récupération des informations complètes, fermeture du programme.");  
 process.exit();  
 } else if (data !== "\r\n" && data !== "\n") {  
 console.log('Vous avez dit: ' + data);  
 }  
});

Vous pouvez à présent lancer votre programme afin de mieux comprendre son fonctionnement.

1. Jouons tous ensemble à un petit jeu

Maintenant vous allez créer petit jeu !

Peut-être connaissez-vous le jeu des allumettes ? Le principe est assez simple. Vous allez disposer de 13 allumettes et à tour de rôle, vous pourrez retirer entre 1 et 3 allumettes. Celui qui retire la dernière allumette, perd. Ici, vous devez faire une IA qui jouera contre l’utilisateur du programme.



*Pourrez- vous battre votre propre IA ?*

À chaque tour, il faudra demander à l’utilisateur combien d’allumettes veut-il enlever et vérifier que le nombre est bien compris entre 1 et 3.

La première étape consistera à déclarer vos variables globales qui vous serviront tout au long de ce programme. La variable sera celle qui aura pour valeur "process.stdin" afin de récupérer l’entrée standard. Il est conseillé d’avoir votre nombre restant d’allumettes qui soit également dans une variable. Vous aurez à vous en servir à de nombreuses reprises.

La boucle principale aura lieu dans la fonction anonyme de la méthode "on()" de la variable de l’entrée standard.

let entreeStandard = process.stdin;

entreeStandard.setEncoding("utf-8");

let nb\_allumette = ...;

entreeStandard.on("data", (data) => {

...

});

Pensez donc à bien découper votre code en fonctions qui seront appelées dans cette boucle. Pour commercer, il faudra faire une fonction qui va afficher les allumettes restantes. Faites-le via une fonction, qui prendra en paramètre le nombre d’allumettes à afficher. Par la suite, une fonction qui s’occupe du tour de l’IA et pour terminer, une qui annonce le gagnant et quitte le programme.

Une étape importante, c’est l’analyse. C’est-à-dire la récupération et vérification de valeurs afin de pouvoir les exploiter. Voici quelques exemples d’analyses :

* Vérifier qu’un message n’est pas vide
* Si le message est bien un nombre
* Vérifier la valeur d’allumettes à enlever, qui doit être comprise entre 1 et 3
* ...

Pensez bien à vérifier le nombres d’allumettes restantes pour savoir si votre programme doit se terminer ou non. Pour quitter un programme, il vous faut utiliser "process.exit()".

Vous souhaiterez sûrement que votre IA, dans un premier temps, fasse de l’aléatoire pour prendre X allumettes. En JavaScript, on utilise la [méthode "random()"](https://www.w3schools.com/js/js_random.asp), qui génère un nombre à virgule entre 0 et 1, de l’objet Math. Mais pour plus de précision sur le nombre généré, on va utiliser la [méthode floor()](https://www.w3schools.com/jsref/jsref_floor.asp), qui transforme le nombre en entier, de Math, encore une fois.

Pour définir vos limites, 1 à 3 inclus, vous devez multiplier le résultat de "floor()" par limite supérieur - 1 et vous devez ajouter la borne inférieure au résultat. Ce qui nous donne ceci :

Math.floor(Math.random() \* 2) + 1

1. Conclusion

Voilà c’est la fin, Monseigneur Node est fier de votre travail et est heureux de voir que la relève est assurée ! Cependant, Monseigneur Node est sûr qu’il reste encore plein d’autres aspects à découvrir ! N’hésitez pas à vous renseigner par vous-même.

En attendant, voici des perspectives d’amélioration que vous pouvez apporter à votre IA :

* Tester différents algorithmes
* Créer un algorithme qui peut vous battre
* Faire un rendu graphique du jeu



Bravo !